**CAPSTONE 2**

**Firstly, we import the required libraries:**

import numpy as np

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

from sklearn.metrics import accuracy\_score, confusion\_matrix

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from imblearn.ensemble import BalancedRandomForestClassifier

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestClassifier, HistGradientBoostingClassifier

from colorama import Style, Fore

red = Style.BRIGHT + Fore.RED

blu = Style.BRIGHT + Fore.BLUE

mgt = Style.BRIGHT + Fore.MAGENTA

gren = Style.BRIGHT + Fore.GREEN

blk = Style.BRIGHT + Fore.BLACK

res = Style.RESET\_ALL

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from xgboost import XGBClassifier

from sklearn.metrics import accuracy\_score

**Now, Read the CSV file and load it into a DataFrame:**

df = pd.read\_csv("Passanger\_booking\_data.csv")

**1. Find any duplicate rows in dataset and remove all the duplicated if found?**

To identify duplicate rows we use the duplicated() method and to remove the duplicate rows use drop\_duplicates() method.

* print("Total number of duplicate rows in data")

print(df.duplicated().sum())

df=df.drop\_duplicates()

print("The shape of the data after removing the duplicate rows")

print(df.shape)

A screenshot of a computer program

Description automatically generated

# 2. Create a pie chart showing count of completed booking and incomplete booking. Is the dataset balanced or imbalanced.

To create a pie chart showing the count of completed bookings and incomplete bookings, and to determine whether the dataset is balanced or imbalanced, we can use the following code:

* import matplotlib.pyplot as plt

booking\_counts = df['booking\_complete'].value\_counts()

print("Total bookings count: ",booking\_counts.sum())

labels = ['Incomplete Booking [0]', 'Completed Booking [1]']

sizes = [booking\_counts[0], booking\_counts[1]]

colors = ['lightcoral', 'lightskyblue']

explode = (0.1, 0)

plt.pie(sizes, explode=explode, labels=labels, colors=colors,

autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.title('Booking Completion Distribution')

plt.show()

# Determine if the dataset is balanced or imbalanced

if booking\_counts[0] == booking\_counts[1]:

print("{blk}The dataset is balanced.")

else:

print("The dataset is imbalanced.")

A screenshot of a computer

Description automatically generated

# 3. Create a bar graph showing booking completed through various sales channel?

* # Group data by sales channel and calculate the count of completed and incomplete bookings

sales\_channel\_data = df.groupby('sales\_channel')['booking\_complete'].value\_counts().unstack().fillna(0)

# Create a bar graph

plt.figure(figsize=(10, 6))

sales\_channel\_data.plot(kind='bar', stacked=False, color=['lightcoral', 'lightskyblue'])

plt.title('Booking Completion by Sales Channel')

plt.xlabel('Sales Channel')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.yticks(range(0, 35001, 5000)) # Set y-axis ticks from 0 to 35000 with an interval of 5000

plt.tight\_layout()

plt.show()

A screenshot of a computer screen

Description automatically generated

# 4. Create a bar graph showing booking completed on all flight\_days?

* flight\_day\_data = df.groupby('flight\_day')['booking\_complete'].value\_counts().unstack().fillna(0)

plt.figure(figsize=(10, 6))

flight\_day\_data.plot(kind='bar', stacked=False,width = 0.8, color=['lightcoral', 'lightskyblue'])

plt.title('Booking Completion on all flight days')

plt.xlabel('flight day')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**A graph of different colored bars

Description automatically generated with medium confidence**

# 5. Create Pie Chart showing booking\_completed for following features

* features = ['wants\_extra\_baggage', 'wants\_preferred\_seat', 'wants\_in\_flight\_meals']

labels = ['0', '1']

colors = ['lightcoral', 'lightskyblue']

explode = (0.1, 0)

plt.figure(figsize=(15, 5))

for idx, feature in enumerate(features, 1):

plt.subplot(1, 3, idx)

values = df[feature].value\_counts()

sizes = [values[0], values[1]]

plt.pie(sizes, explode=explode, labels=labels, colors=colors,

autopct='%1.2f%%', shadow=True, startangle=140)

plt.title(feature)

plt.axis('equal')

plt.tight\_layout()

plt.show()

A screenshot of a computer screen

Description automatically generated

# 6. Convert following categorical features using one hot encoding:

* df = pd.get\_dummies(df, columns=['sales\_channel', 'trip\_type', 'flight\_day', 'booking\_origin'],drop\_first = True)

A screenshot of a computer

Description automatically generated

# 7. Normalize following Numeric features:

# 'num\_passengers', 'length\_of\_stay', 'flight\_hour', 'flight\_duration

* # Specify the numeric features

numeric\_features = ['num\_passengers', 'length\_of\_stay', 'flight\_hour', 'flight\_duration']

# Initialize the StandardScaler

scaler = StandardScaler()

# Apply standardization to the numeric features

df[numeric\_features] = scaler.fit\_transform(df[numeric\_features])

df.head()

A screenshot of a computer

Description automatically generated

# 8. Create Training and Testing Dataset (Take 12% as test dataset)

# a. Use following Algorithms to train on the dataset:

# i. 'Logistic Regression', ii. 'KNN', iii. 'SVM',

# iv. 'Naive Bayes', v. 'Decision Tree Classifier',

# vi. 'Random Forest Classifier', vii. 'XgBoost'

* # Specify the features and target

features = df.drop(columns=['booking\_complete'])

target = df['booking\_complete']

# Split the dataset into training and testing sets (88% training, 12% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.12, random\_state=42)

# Initialize the StandardScaler and scale the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Initialize and train the models

models = {

'Logistic Regression': LogisticRegression(),

'KNN': KNeighborsClassifier(),

'SVM': SVC(),

'Naive Bayes': GaussianNB(),

'Decision Tree Classifier': DecisionTreeClassifier(),

'Random Forest Classifier': RandomForestClassifier(),

'XgBoost': XGBClassifier()

}

for model\_name, model in models.items():

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'{model\_name} Accuracy: {accuracy:.4f}')

A screenshot of a computer

Description automatically generated

# 9. Find Value of k for which KNN Gives Best Accuracy

* best\_k = None

best\_accuracy = 0.0

for k in range(1, 21):

knn = KNeighborsClassifier(n\_neighbors=k)

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

if accuracy > best\_accuracy:

best\_accuracy = accuracy

best\_k = k

print(f'Best k: {best\_k} with accuracy {best\_accuracy:.4f}')

A screenshot of a computer

Description automatically generated

# 10. Print all evaluation metrics for all the Algorithms(find it on Test Data) i.e Precision, Recall, Accuracy and F1 Score, Confusion Matrix

* from sklearn.metrics import precision\_score, recall\_score, accuracy\_score, f1\_score, confusion\_matrix

# Initialize and train the models

models = {

'Logistic Regression': LogisticRegression(),

'KNN': KNeighborsClassifier(),

'SVM': SVC(),

'Naive Bayes': GaussianNB(),

'Decision Tree Classifier': DecisionTreeClassifier(),

'Random Forest Classifier': RandomForestClassifier(),

'XgBoost': XGBClassifier()

}

for model\_name, model in models.items():

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

cm = confusion\_matrix(y\_test, y\_pred)

print(f'{model\_name} Evaluation Metrics:')

print(f'Precision: {precision:.4f}')

print(f'Recall: {recall:.4f}')

print(f'Accuracy: {accuracy:.4f}')

print(f'F1 Score: {f1:.4f}')

print(f'Confusion Matrix:\n{cm}\n'

A screenshot of a computer

Description automatically generated